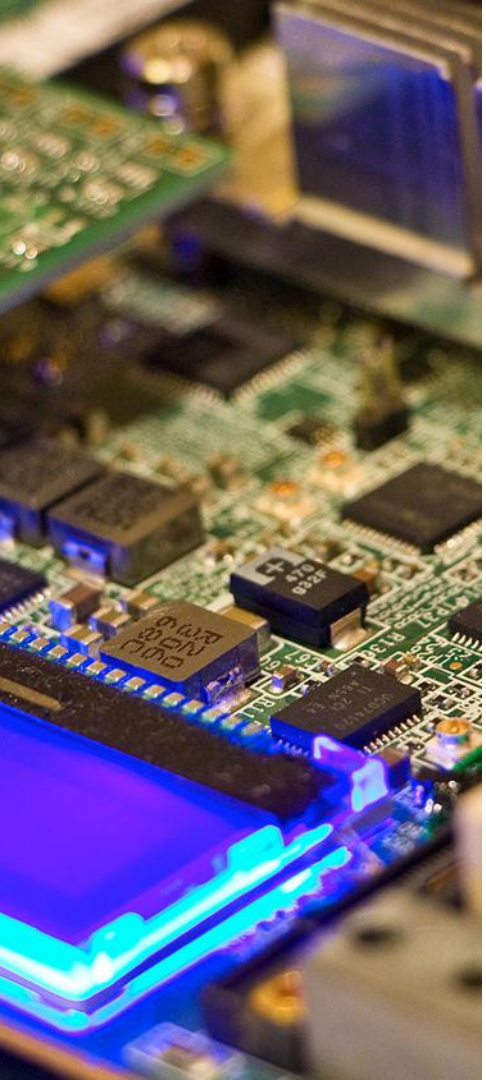
Upstreaming, Downstreaming, “Sidestreaming”... How can AOSP based projects work together?

Presented by

Bernhard “Bero” Rosenkränzer
Tech Lead, Android Engineering
Linaro

Date

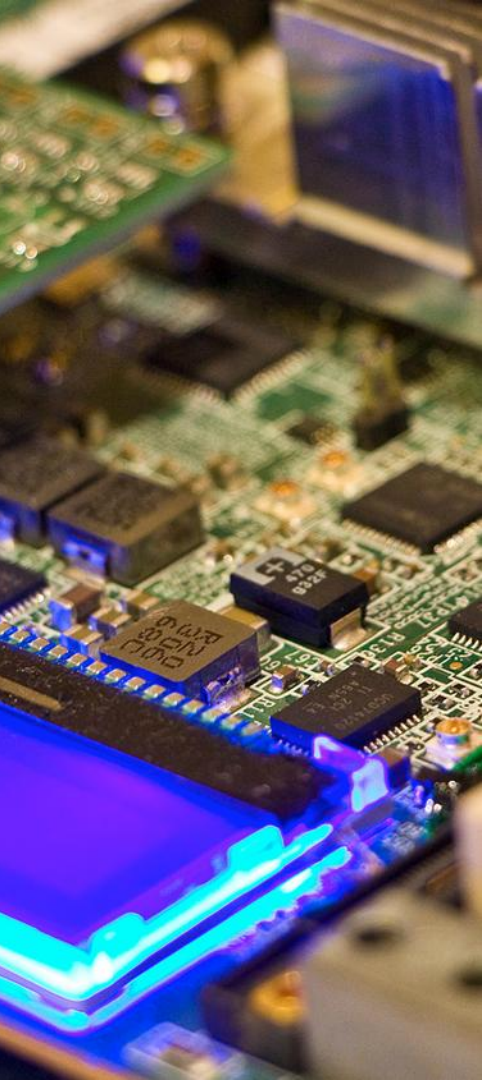
Tue, Mar 24. 2015
Android Builders Summit



The problem...

- There are lots of AOSP based projects out there -- many with patches that would be useful to others.
- Even generally useful changes often don’t get upstreamed.

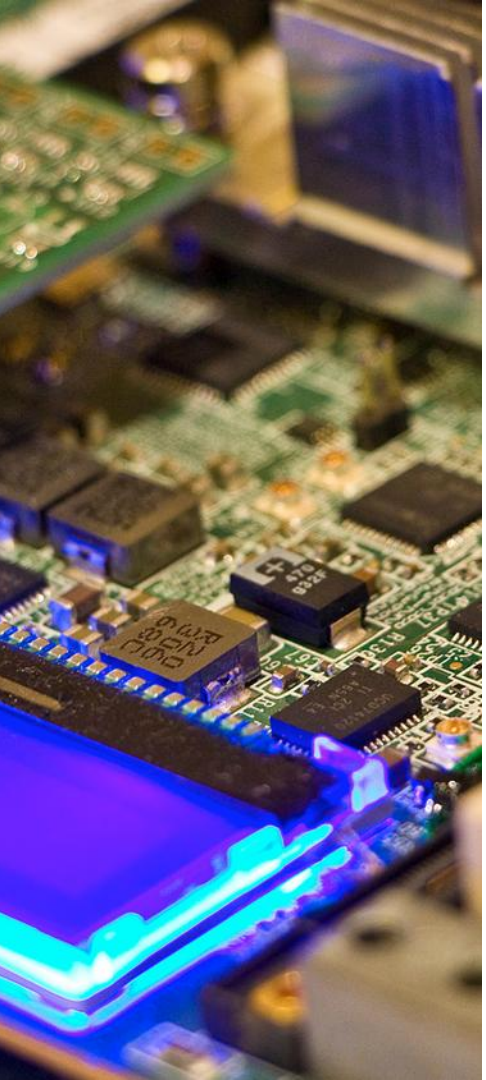
Linaro



The problem...

- Project B usually doesn’t know what Project A is doing or has done
- Most projects don’t make an effort of getting their changes included in other projects - usually a time issue.

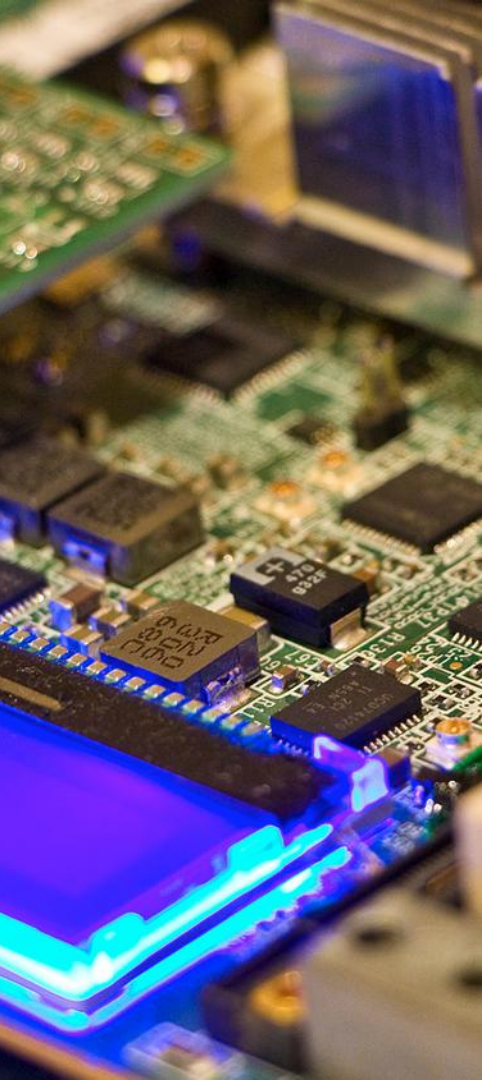
Linaro



The Linaro AOSP tree

- The Linaro AOSP tree (http://android.git.linaro.org/) is yet another AOSP derived OS
- Its primary purpose is to serve as a base for “real” products, and a testing ground for patches to be upstreamed.

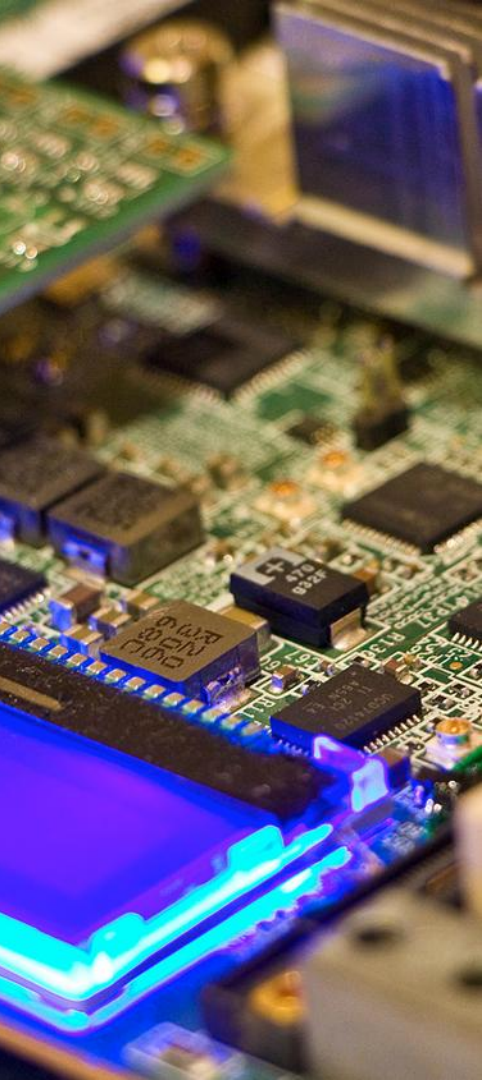
Linaro



Earlier situation

- All our work was committed to Linaro branches on android.git. linaro.org - branches were rebased when new upstream AOSP releases happened.

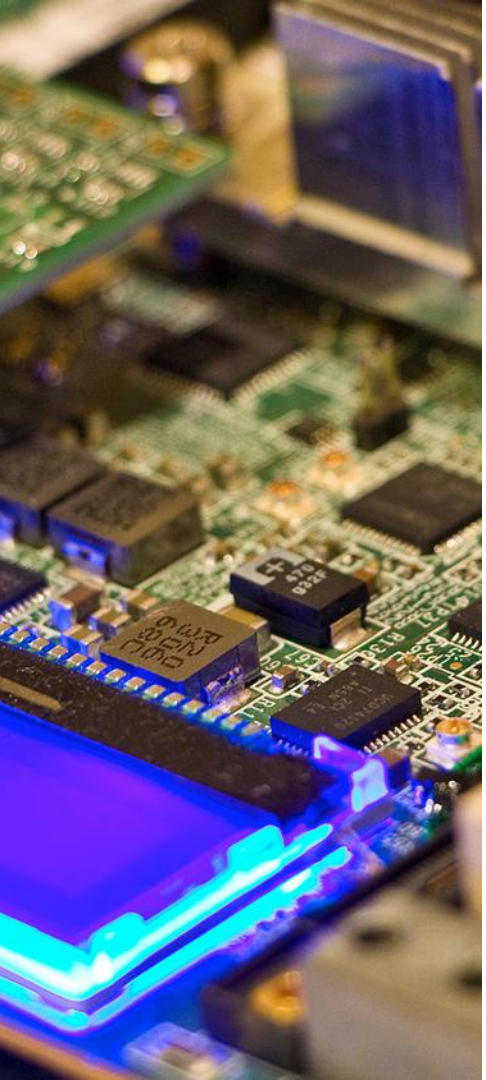
Linaro



What worked well

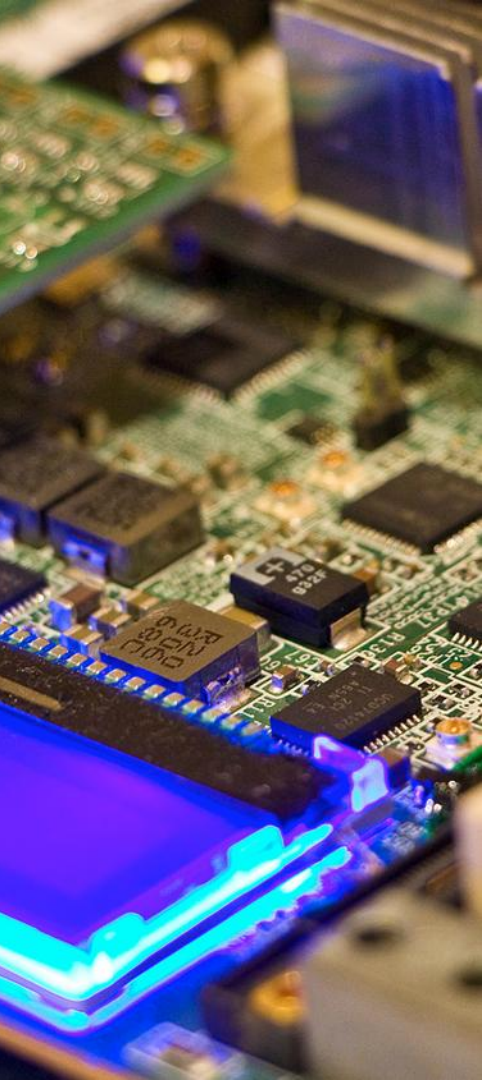
- Very easy for us to check out an up to date repository
- No additional tools required
- Essentially the same workflow as upstream

Linaro



What didn’t work that well

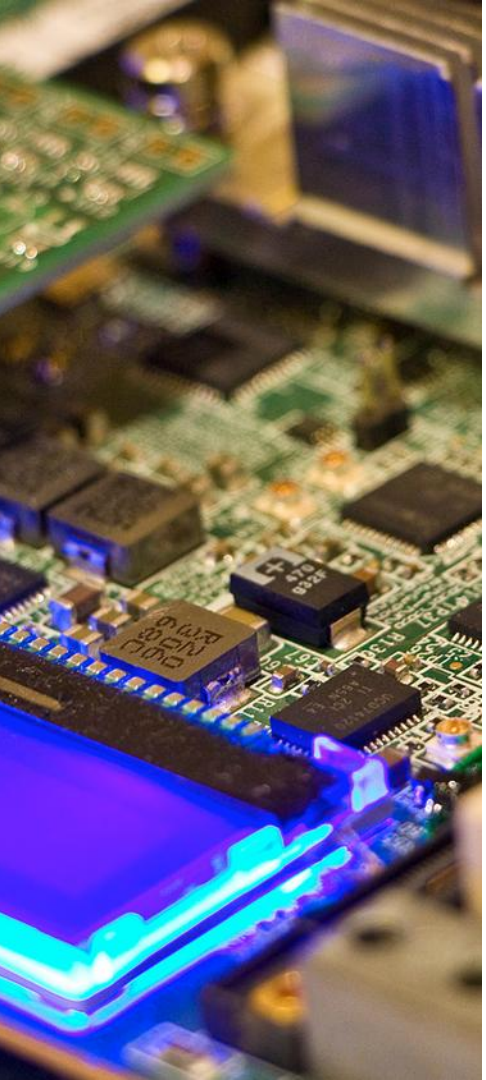
- It was hard to track inside the repositories what patches have been upstreamed, what patches have been submitted upstream but not yet accepted, and what patches were not meant for upstreaming



What didn't work that well

- It was hard for other projects (members' product teams, AOSP based Open Source projects like CyanogenMod and Replicant, …) what patches we're applying, and to determine if they have the complete patchset

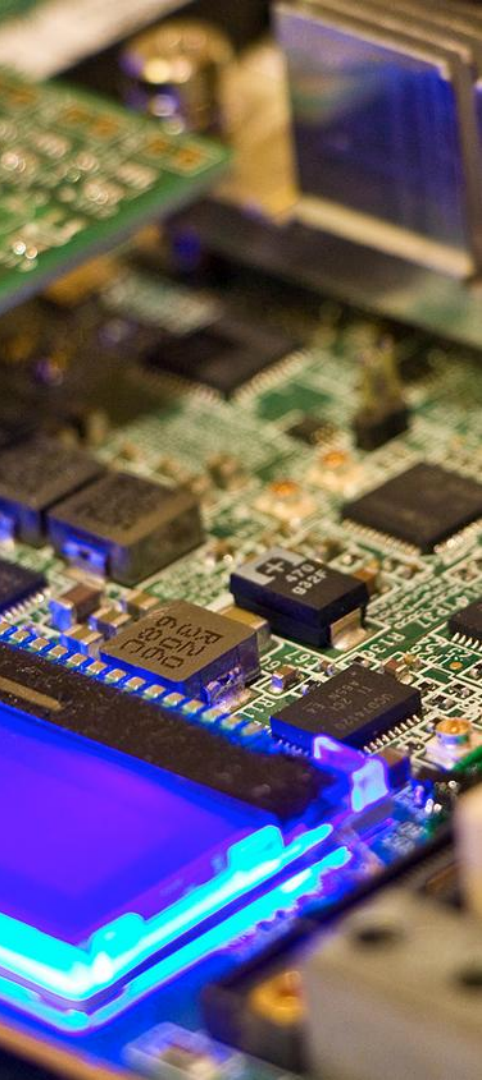
Linaro



Goal: Tracking upstreaming status

- It would be useful to know immediately where our builds differ from upstream, and to make sure all patches that are useful upstream have been submitted...

Linaro

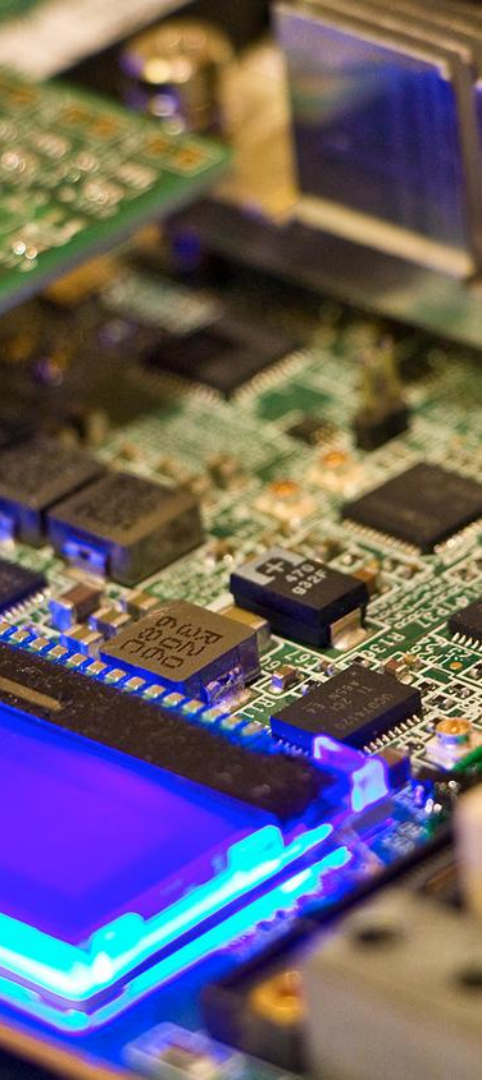


Goal: Tracking upstreaming status

- Many AOSP based projects, including upstream AOSP and Linaro, use Gerrit for code review -- can we just pull patches submitted there from Gerrit?

  Turns out we can, after writing a few scripts.

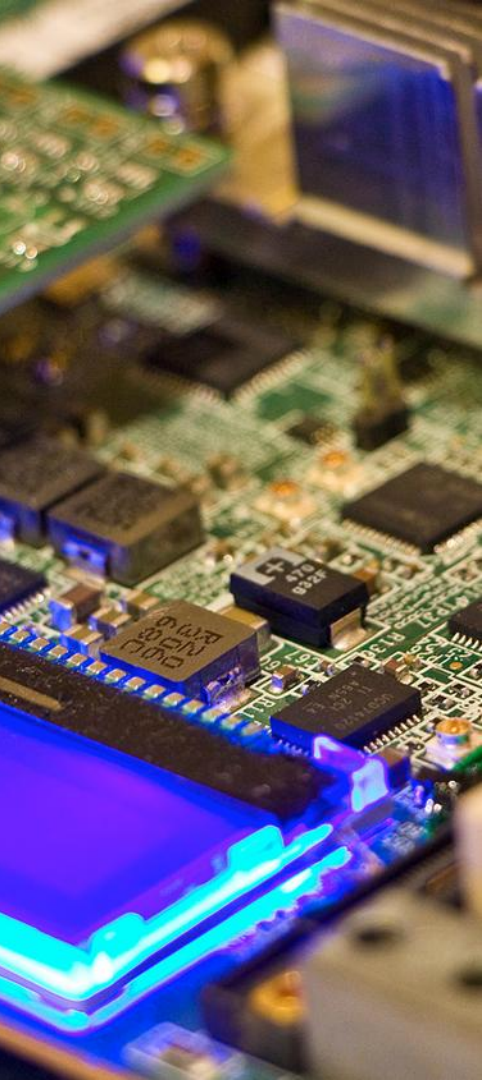
Linaro



Goal: Tracking upstreaming status

- `apply bionic 109026/1`
  Applies revision 1 of the patch submitted to upstream Bionic with submission number 109026. Anyone can look up details in upstream Gerrit: https://android-review.googlesource.com/109026

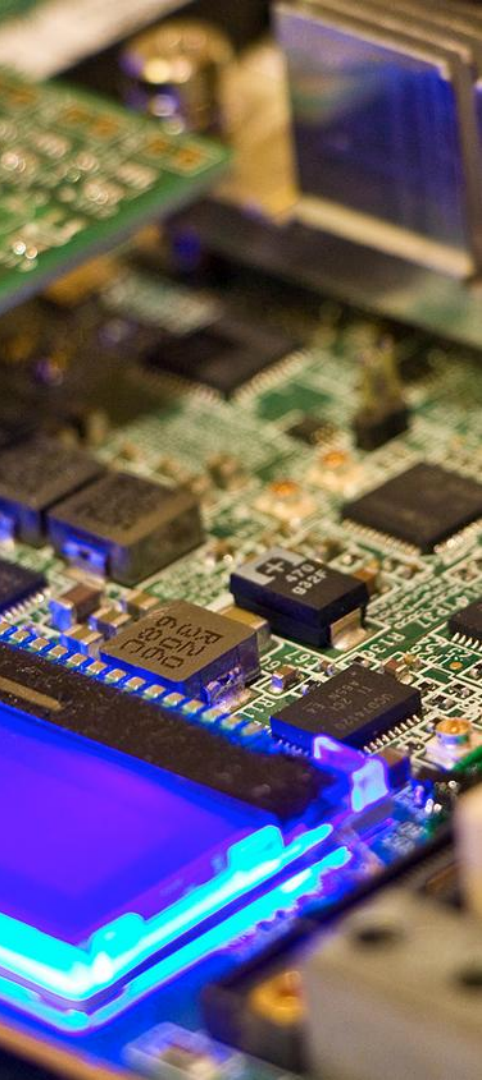
Linaro



Goal: Tracking upstreaming status

- `apply bionic 109026/1`
  Once the patch has been applied upstream, `apply` will fail -- showing us it's safe to remove the patch from the list and "forget" about it.

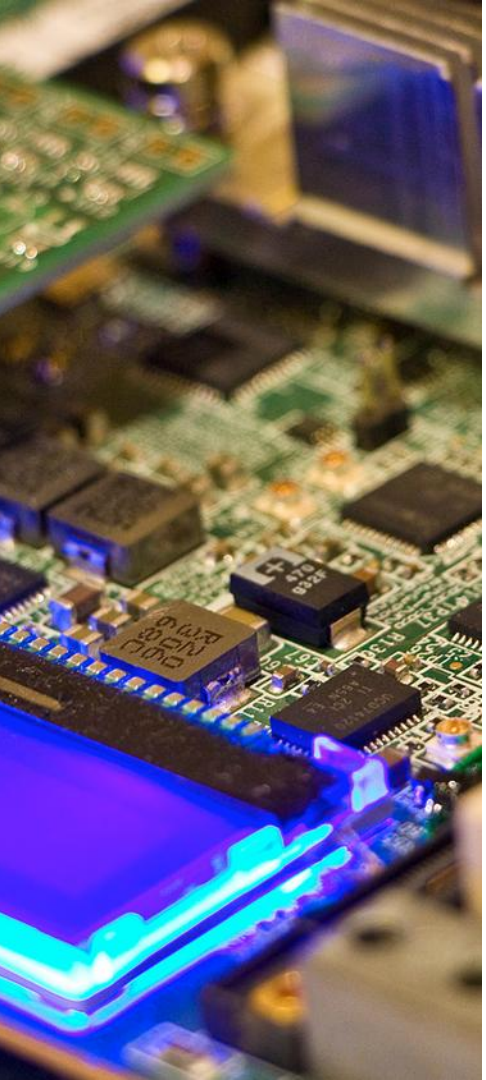
Linaro



Goal: Tracking upstreaming status

- `apply bionic 109026/1`
  If the patch conflicts with upstream changes (esp. on master branch), `apply` fails -- and we know it's time to submit a new revision.

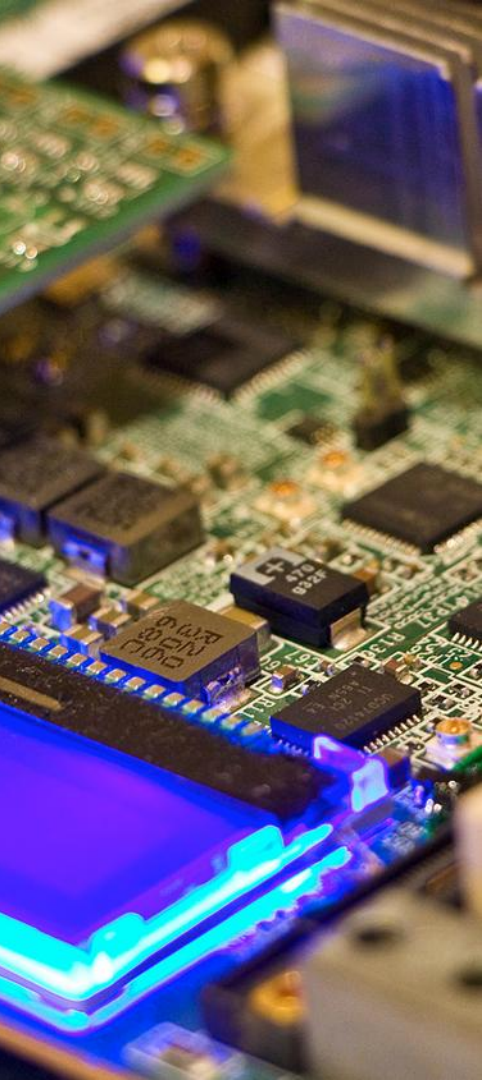
Linaro



Goal: Documenting local patches

- `apply --linaro system/extras 15343/1`
  Applies revision 1 of submission 15343 to system/extras in Linaro's local Gerrit.
  Anyone can look up details in local Gerrit:
  https://review.android.git.linaro.org/15343

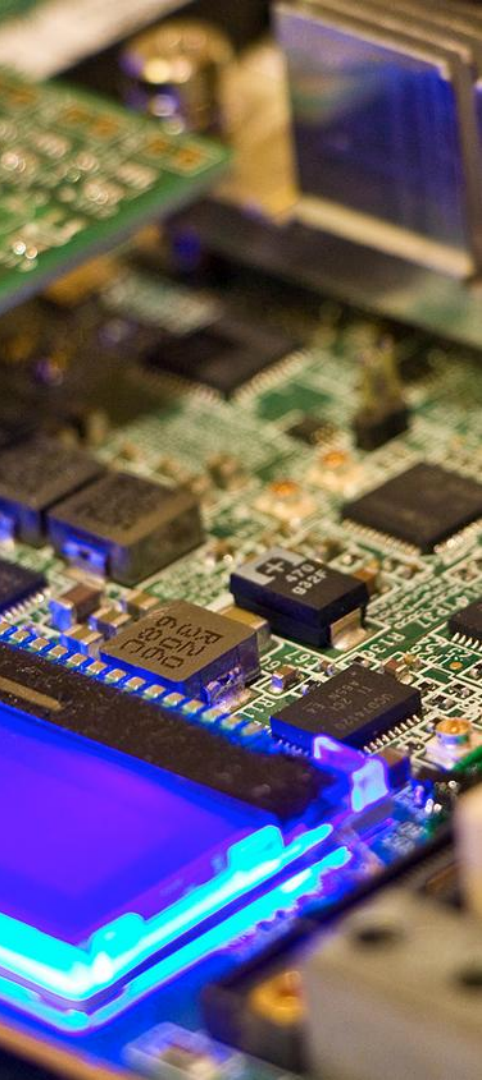
Linaro



Possible future feature

- `apply --replicant hardware/ril 1234/5`
- `apply --cyanogen bionic 12345/6`
- …
- Pulling in interesting patches from other AOSP derived projects should be easy (as long as they use Gerrit) -- having the submission number etc. in the script always makes it easy to look up details without having to write extra documentation that is in danger of getting out of sync with reality.

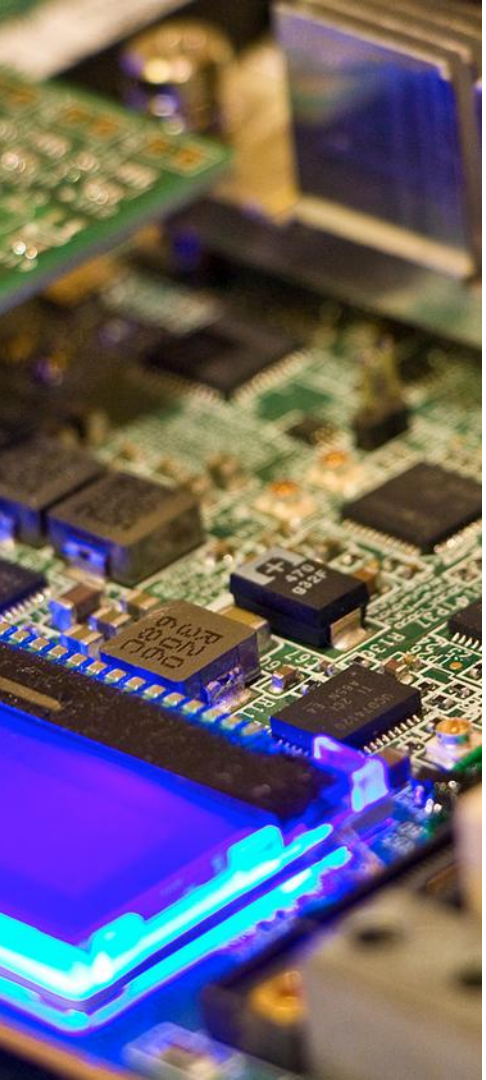
Linaro



Another useful addition

- `cherrypick bionic 12345678`
  Cherry-picks patch 12345678 from another git branch (Linaro specific branches where they're still useful, or pulling a patch from AOSP master into a release-based build, etc. …)

Linaro

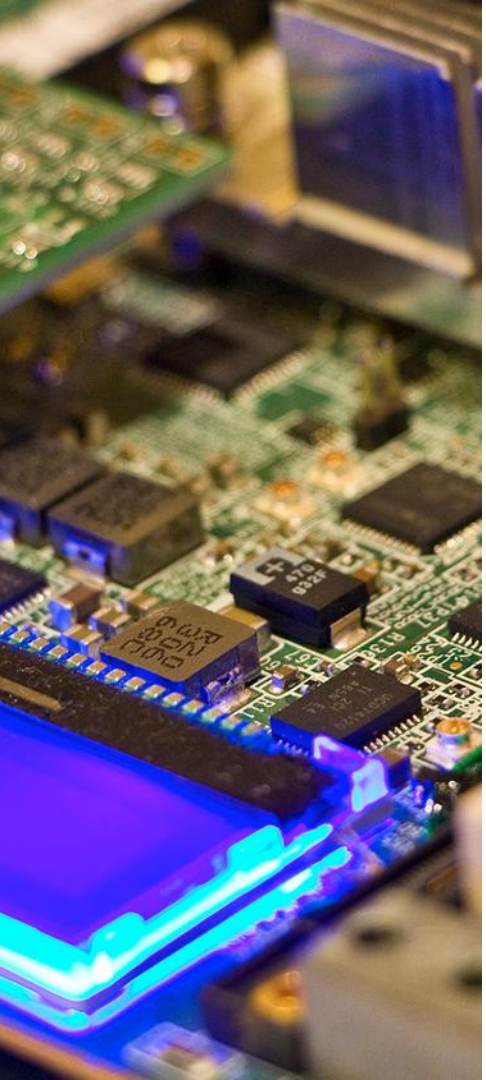


What about extra trees?

There’s no device/linaro/* repositories in AOSP -- so using “`apply`” to pull patches into them is not useful.

But repo has a not-so-well-known feature for this: local manifests.

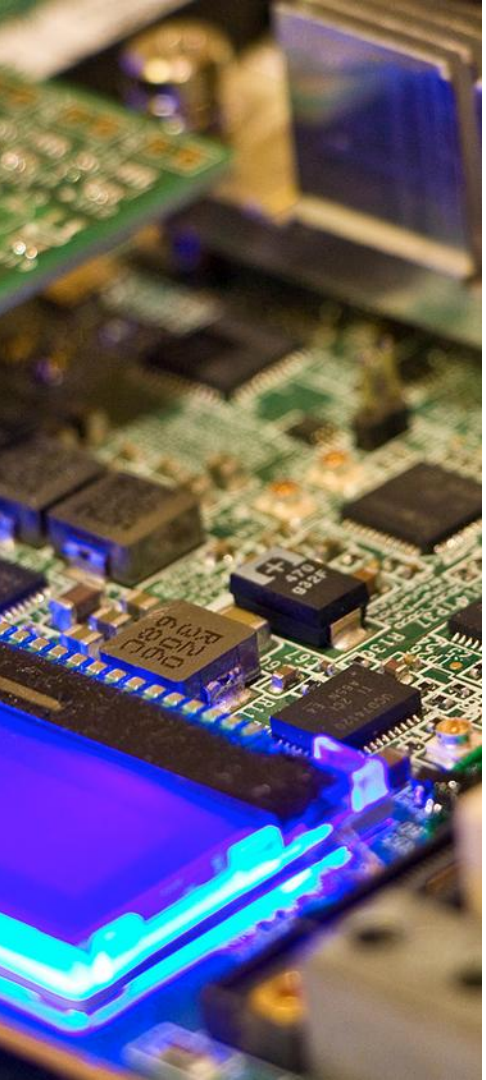
Linaro



Local Manifests

Without local manifests: manifest files have to be modified on a separate Linaro branch for every AOSP version we're supporting -- and we don't automatically pull in modifications from upstream (e.g. added project in upstream master branch goes missing)

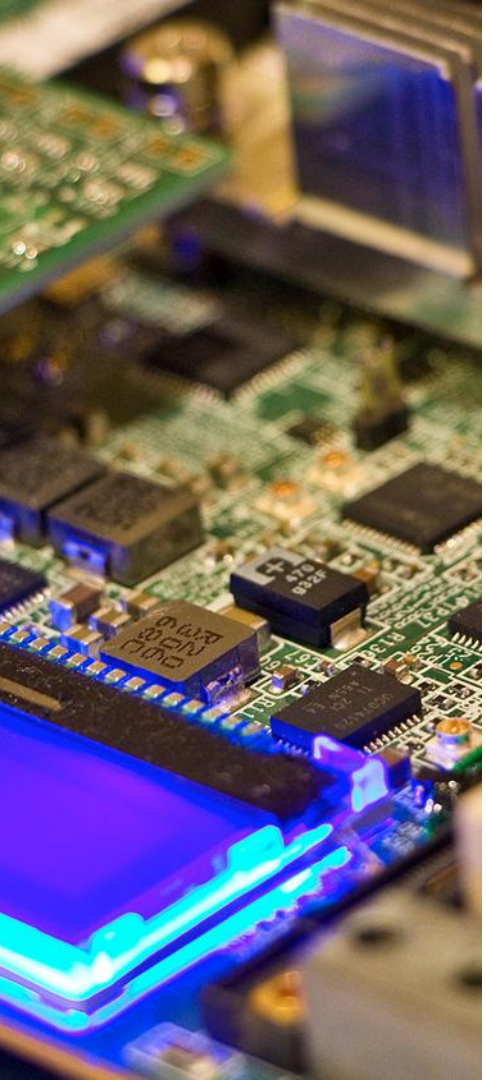
Linaro



Local Manifests

With local manifests: repository is checked out from upstream manifest files -- an extra manifest file is pulled in to add local trees (`device/linaro/*`) and override bits we want to pull from elsewhere.

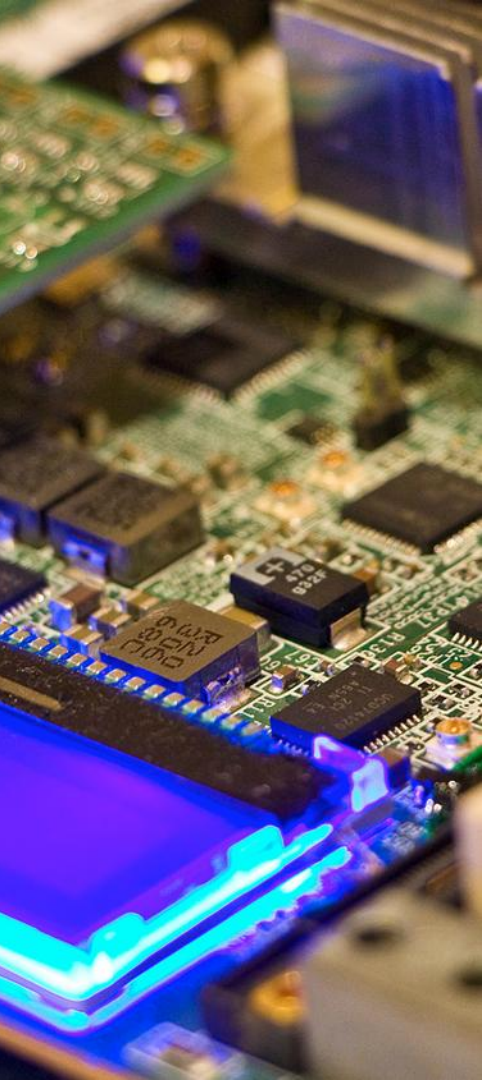
Linaro



Local Manifests

Other projects can pull in our local_manifests to add support for all devices supported by our tree (or our toolchains, or other extras) to their own modified AOSP tree, not necessarily based on ours

Linaro



Local Manifests: example

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
  <remote name="linaro-android"
          fetch="git://android.git.linaro.org/" />
  <!-- Pull in support for extra devices -->
  <project path="device/linaro/vexpress"
           name="device/linaro/vexpress"
           remote="linaro-android" />
  <!-- Drop libjpeg for libjpeg-turbo -->
  <remove-project name="platform/external/jpeg" />
  <project name="platform/external/jpeg"
           path="platform/external/libjpeg-turbo"
           remote="linaro-android" />
</manifest>
```

Linaro

Questions? Feature requests?

- Speak up now or mail us later...

Linaro